Министерство образования Республики Беларусь

Учреждение образования

Белорусский государственный университет информатики и радиоэлектроники

Лабораторная работа «Игра Тетрис»

Выполнила студентка
учебной группы 450501

Попеня В.Н.

Проверил преподаватель

Кухарчук И.В.

Минск 2016

Реализуемая игра: тетрис.

Одна из самых знаменитых компьютерных игр, суть которой заключается в том, чтобы из комбинации случайно появляющихся фигур, составить полностью заполненную строку. Такие строки убираются, и за каждую убранную строку прибавляется один балл. Фигуры, которые не составили заполненную линию, собираются на игровом поле. Когда добавление новых фигур невозможно, игра считается законченной. Задача игры: набрать как можно больше очков.

Главное меню содержит:

- Дополнительное меню
- Сложность (от уровня сложности зависит скорость падения фигур и их количество.)
- Играть с выбранной сложностью. По умолчанию выбирается «Нормально»
- Выход из игры

Рис.1 Главное меню

Дополнительное меню:

- Выбор типа воспроизводимой игры: последняя или другая (для сортировки игр)
- Игра бота
- Вернуться к главному меню

Рис. 2 Дополнительное меню

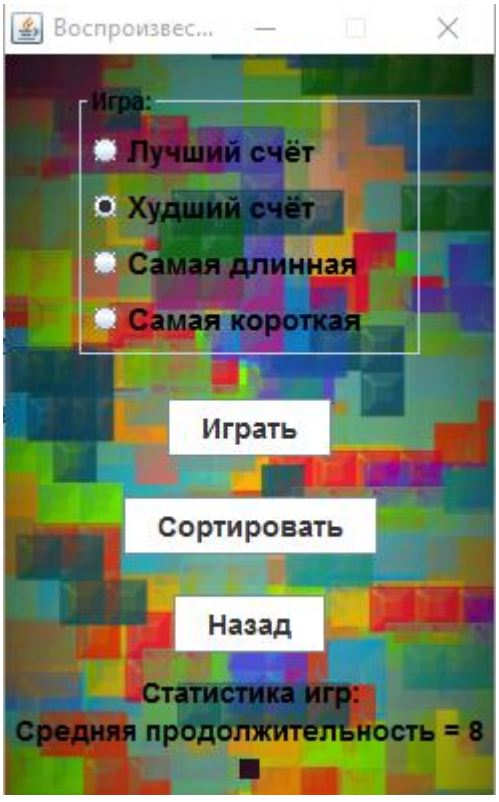


Рис. 3 Меню воспроизведения игр

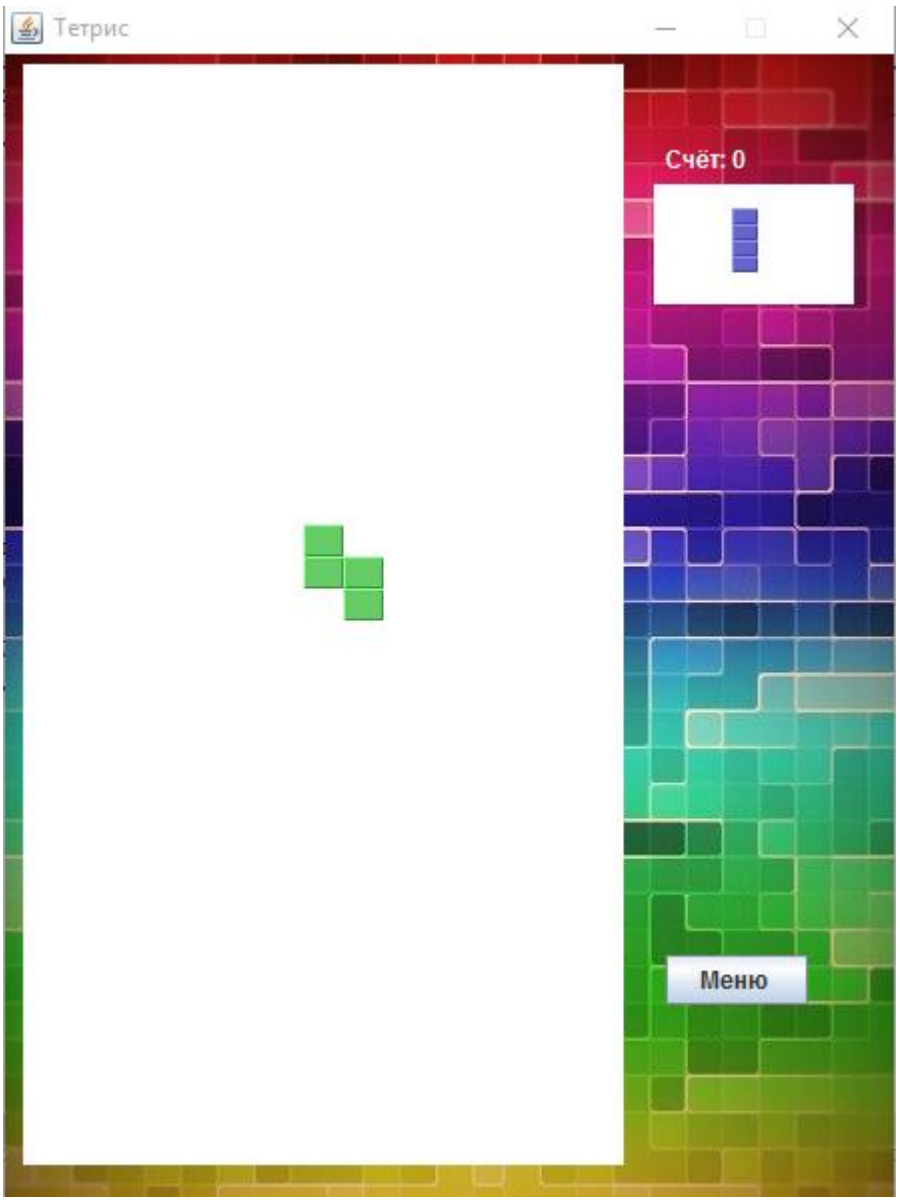


Рис.4 Окошко игры (ур. сложности - нормально)

В окне “Тетрис” реализована логика игры, при нажатии на клавиши:

«право», «лево» - фигура вертикально передвигается по полю

«верх», «низ» - фигура вращается

«пробел» - фигура «падает»

«D», «d» - фигура передвигается на одну линию вниз

«P», «p» - игра приостанавливается/запускается.

Диаграмма классов представлена в приложении А.

В классе LogicGame осуществляется основная логика игры и выполняются основные операции над падающими фигурами. Это позволяет другим

классам при необходимости обращаться лишь к конкретному методу класса LogicGame, который сам определяет необходимые данные и действия для совершения заданной задачи. Это явление можно наблюдать на примере класса KeyboardHandler. При нажатии на клавишу выбирается конкретный метод из класса LogicGame, в соответствии с которым фигура передвигается в заданном направлении либо вращается, либо игра приостанавливается (возобновляется). Как видно из диаграммы классов, универсальность класса LogicGame приводит к его многочисленному использованию, что позволяет добиться точного выполнения поставленных задач.

Классу LogicGame необходимы данные класса Complexity, потому что в зависимости от выбранной сложности изменяется время обновления экрана, ширина и высота игрового поля. Сложность выбирается в окне, создаваемом классом MenuMainWindow. Создание класса Complexity помогает сделать приложение более универсальным, так как поля, содержащиеся в этом классе, позволяют определять свои значения только в нём, а классы, которые использую эти поля, могут и не знать, что в них храниться, и при необходимости изменения этих значений, они будут изменены только один раз в классе Complexity.

В классе Shape осуществляются все необходимые операции над фигурой, так же в нём хранятся координаты как текущей фигуры, так и всех возможных. Благодаря данному классу вся работа с самой фигурой (вращение, определение текущих координат и тд.) сводится только к вызову необходимых методов. Что приводит не только к универсальности данного класса, но и к защите данных о фигуре (её координат, типа) от случайных изменений.

Самогенерируемая документация класса Complexity

Блок схема алгоритма actionPerformed() представлена в приложении Б. Это основная функция класса LogicGame, которая вызывается через каждое срабатывание таймера, и определяет действие, производимое над фигурой.

Разработанная нотация выглядит следующим образом:

Таблица 1

NewGame	Поле, которое записывается только один раз при запуске игры
Тип уровня сложности	Записывается один раз. Бывает: Easy, Normal, Hard
Игровое поле	Значения каждого символа – тип фигуры в данной координате. (табл. 2)
Текущий счёт	Значение поля score

Текущая и следующая фигуры	См. табл. 2

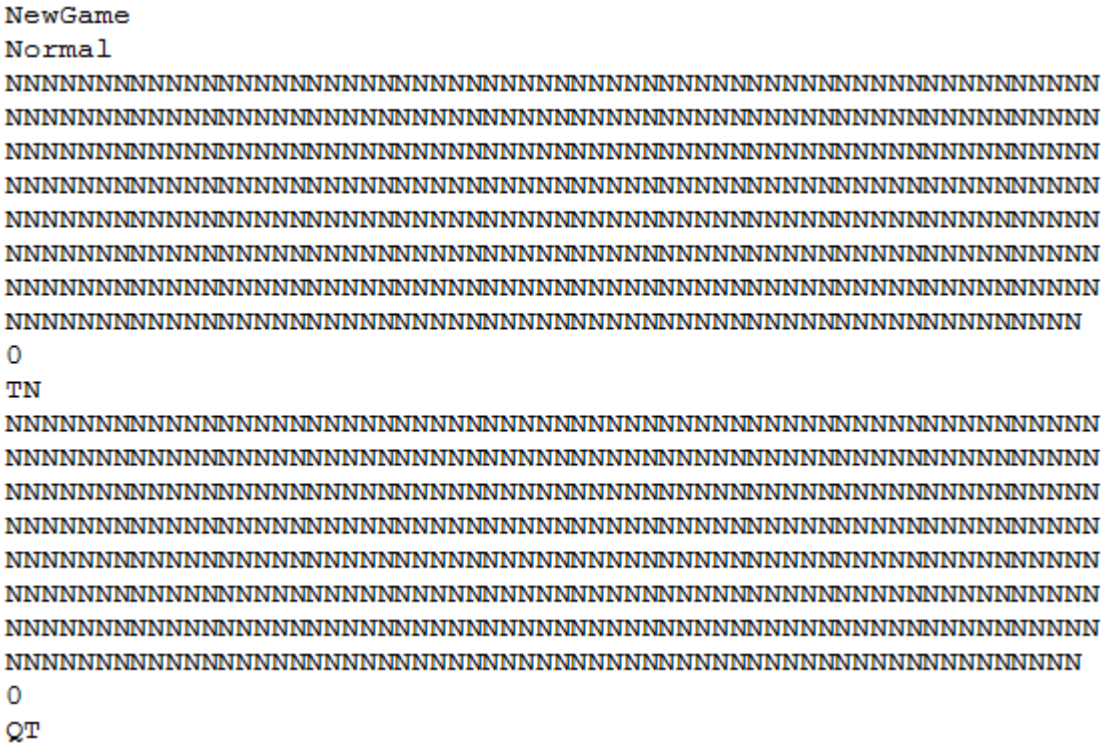


Рис. 5 Скрин файла с игрой

После “NewGame” сообщается информация о сложности воспроизводимой игры. В новой строке записано игровое поле, которое представляет закодированные в символы фигуры игры. После следует счёт игрока. В последней строке следующая и текущая фигуры.

Фигуры закодированы следующим образом:

Таблица 2

NoShape	N
ZShape	Z
SShape	S
LineShape	I
TShape	T
SquareShape	Q
LShape	L
MirroredLShape	M

Данный вид нотации позволяет упростить понимание фигуры до одного символа, позволяет гибко подстраиваться под выбранные сложности, оперативно реагировать на изменение игрового счёта, отображать текущую и

следующую игровые фигуры, что делает процесс воспроизведения игры относительно простым.

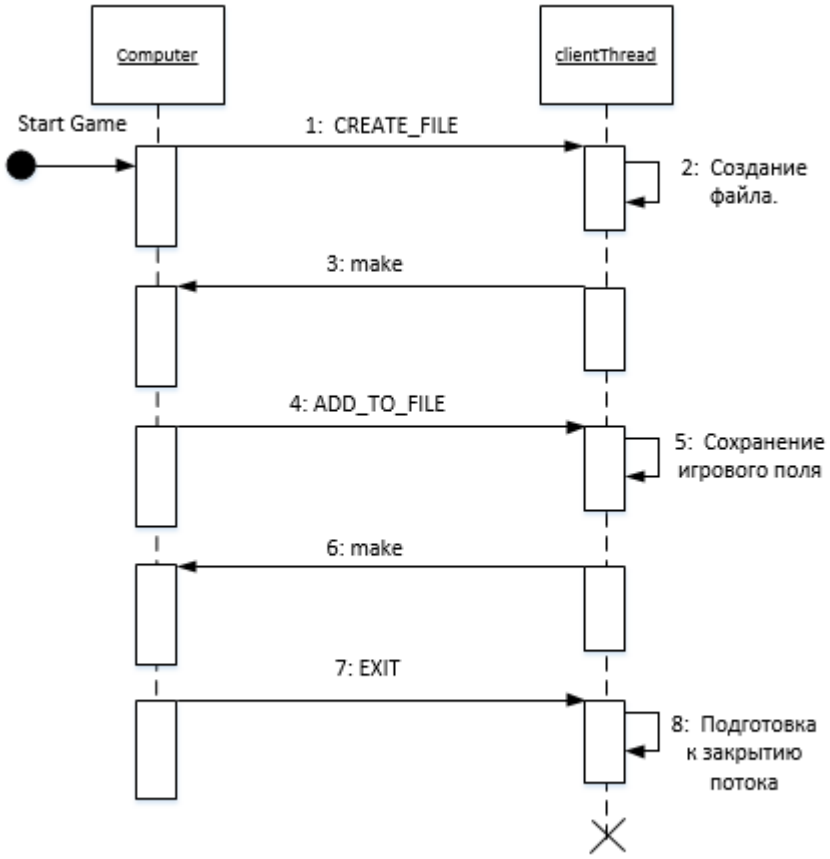


Рис. 5 Диаграмма последовательности клиент-серверного взаимодействия

Описание работы потоков:

0. При нажатии на кнопку играть создаётся поток, который отвечает за сохранение игры.
1. В первый раз в него отправляется команда «CREATE_FILE». Поток создаёт файл с нужным именем.
2. Создание файла.
3. После этого он отправляет главному потоку сообщение «make».
4. Далее отправляется команда «ADD_TO_FILE», при чтении которой поток записывает в файл текущие данные.
5. Сохранение игрового поля.
6. Поток отправляет главному потоку сообщение «make», что говорит об удачном завершении команды.
7. Отправляем сообщение «EXIT» для правильного закрытия используемых ресурсов.
8. Подготовка потока к закрытию.

Создаются ещё два потока для отображения игрового поля и поля следующей фигуры.

Сортировка игры:

Таблица 5

	1000	10000	100000
Scala	3420	11620	69440
Java	396	1504	10094

Приложение А Часть 1(Диаграмма классов)

Пакеты игры:

- Main –пакет с классом запуска игры

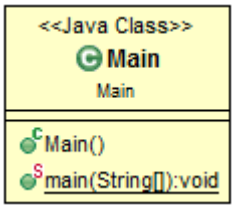


Рис. 1 Класс main

- Bot – пакет с классами, реализующими логику работы бота.
  - RandomBot – класс основной логики бота, работа которого основывается на случайном выборе действия
  - Handler - Обработчик действий, которые определяются в классе RandomBot.

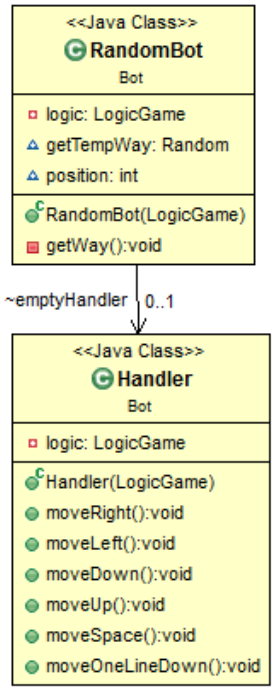


Рис. 2 пакет Bot

- Sort – пакет, в котором содержатся классы для работы с сортировками, а так же работы со статистикой.
  - JavaSort – Сортировка на Java
  - ScalaSort – Сортировка на Scala

- Statistics – класс обработки статистики

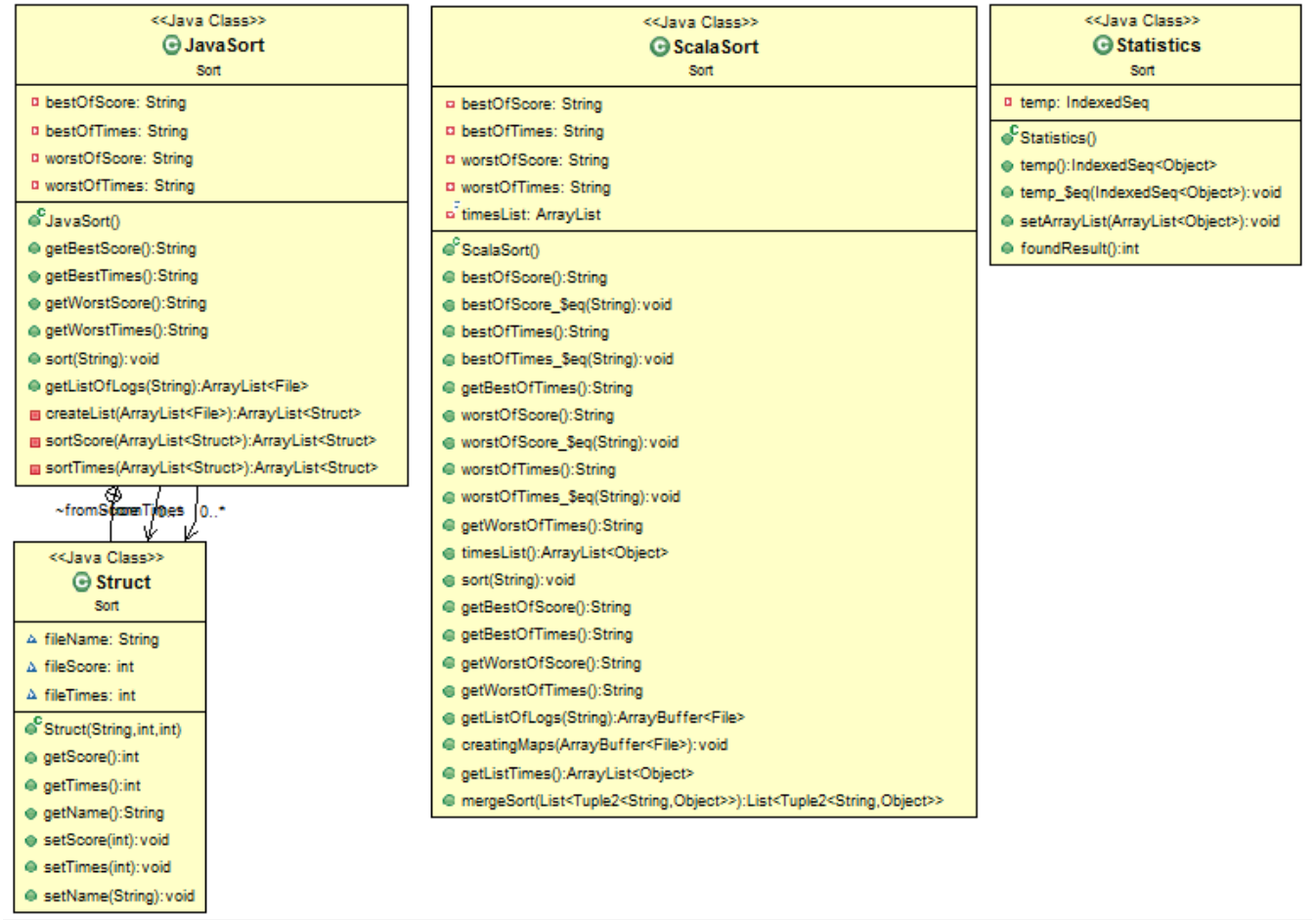


Рис.3 Пакет Sort

- TetrisLogic – пакет с логикой игры
  - KeyboardHandler – обработчик клавиатуры, при нажатии клавиши выбирает необходимое действие.
  - LogicGame – основной класс логики игры, в котором определяется следующая фигура, текущая фигуры, время падения, убираются заполненные линии, вызываются методы классов, отвечающих за replay, производится попытка передвинуть фигу по игровому полю.
  - PaintNextShape – класс, который отображает на экране следующую фигуру.
  - Complexity – класс с информацией о выбранной сложности игры.
  - PaintBoardGame – класс, отображающий игровое поле.
  - Shape – класс, в котором хранится информация о всех возможных фигурах, их координатах и типа, а так же методы работы с ними.

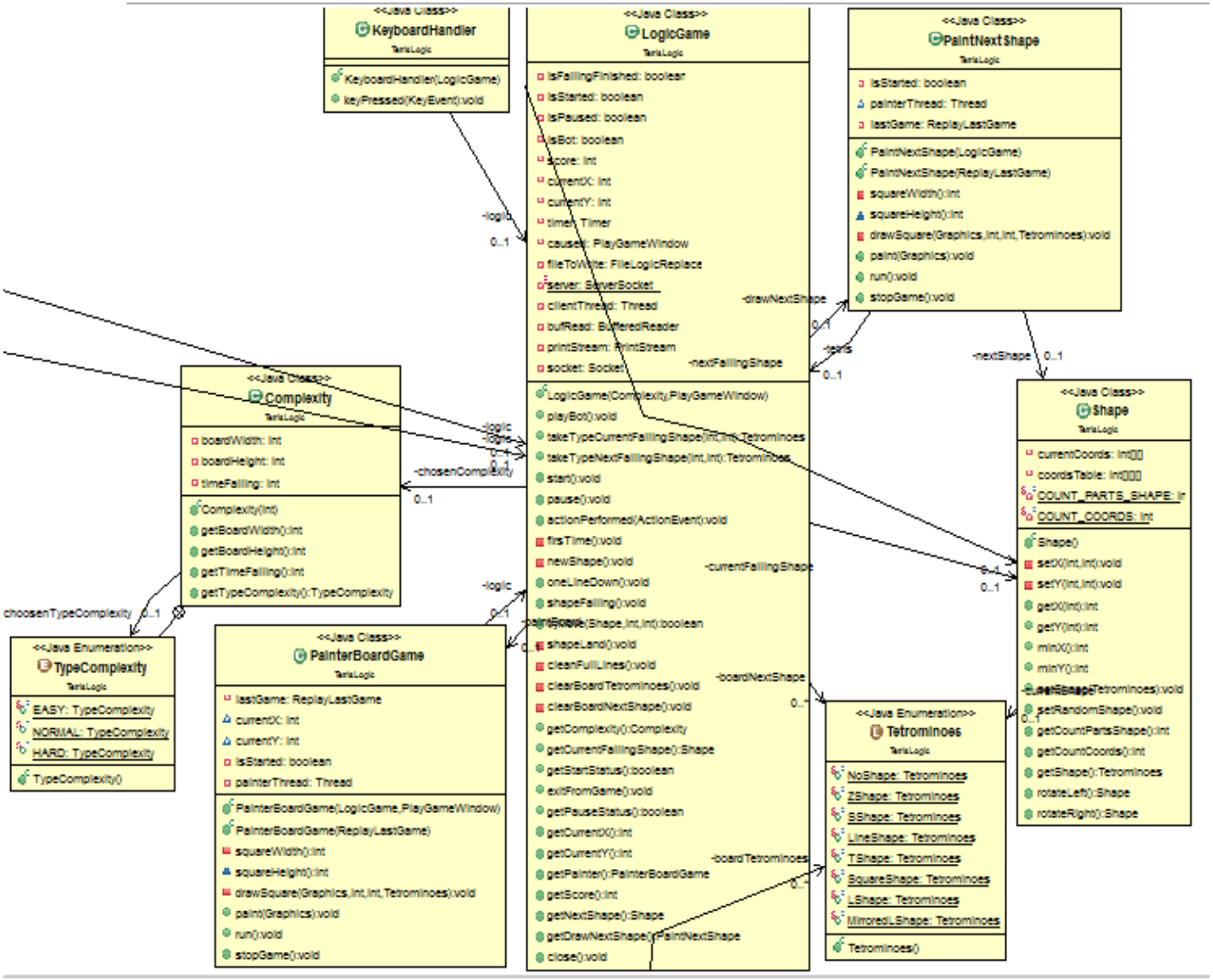


Рис. 5 Пакет TetrisLogic

- FileWorking – пакет с логикой записи в файл
    - TextFile – класс, в котором содержатся методы непосредственной работы с файлом
    - FileLogicReplace – класс, в котором определяется то, что будет записано в файл

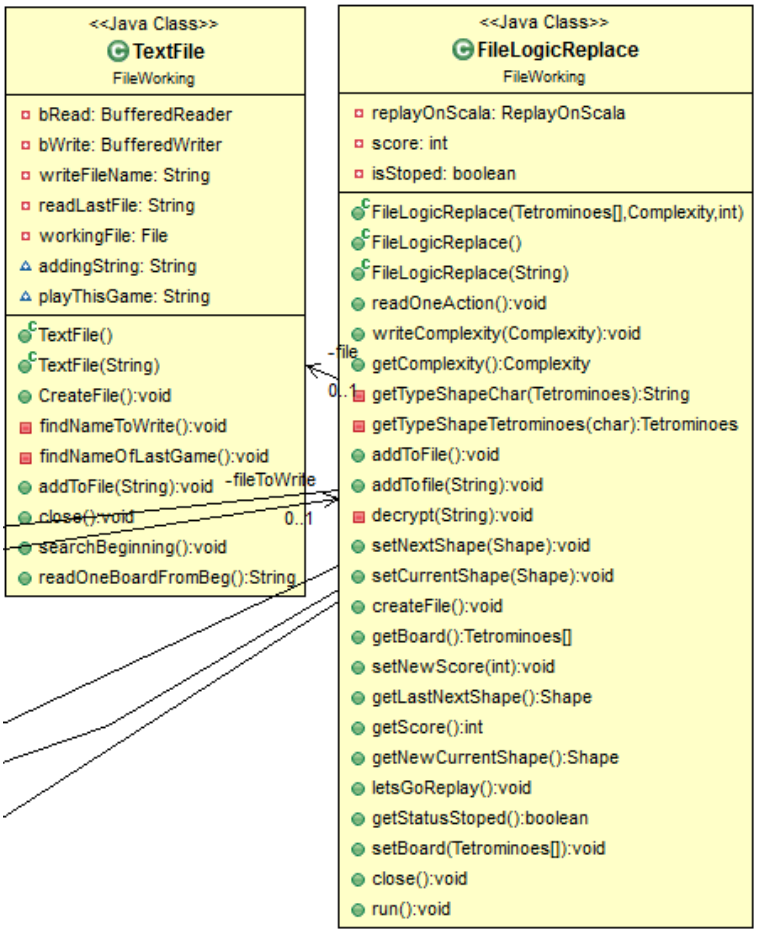


Рис.6 Пакет FileWorking

- Replay – пакет с логикой для replay.
    - ReplayLastGame – воспроизводит последнюю игру, однако если указать в конструкторе при создании объекта имя файла с логами игры, то воспроизведётся эта игра.
    - ReplayOnScala – класс содержит методы для создания читаемой нотации replay.

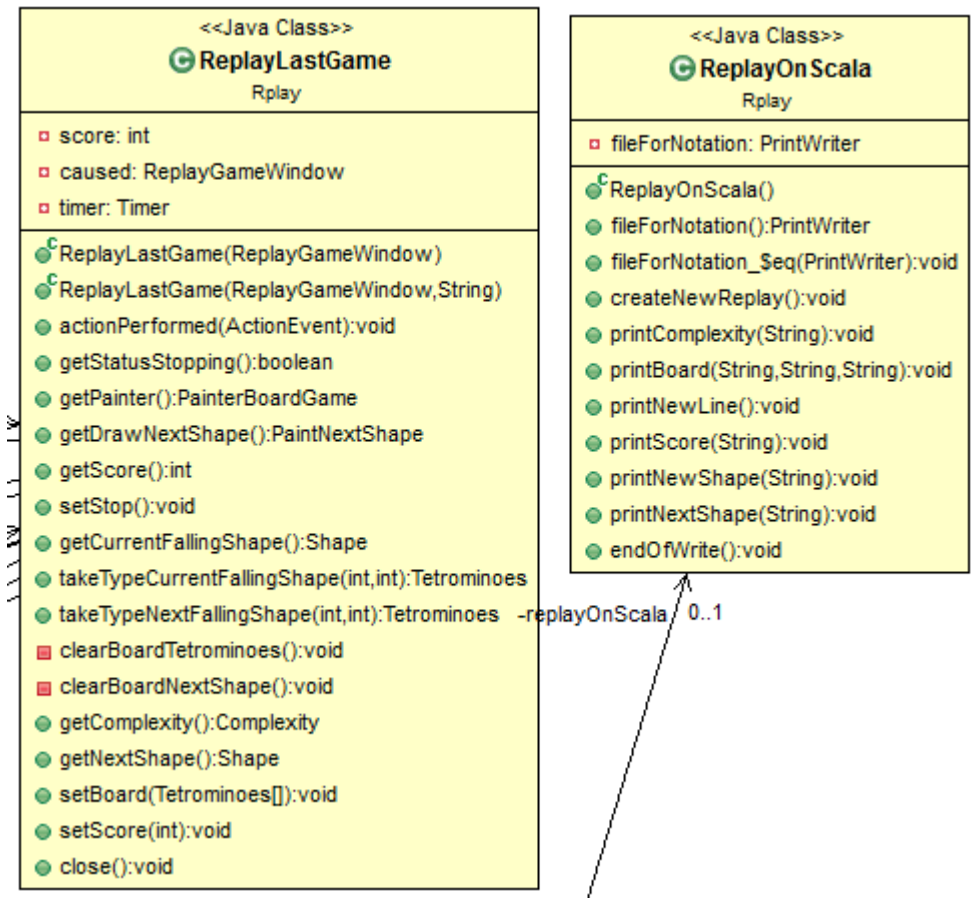


Рис. 7 Пакет Replay

- Windows – все окна игры.
  - SortGameWindow – окно с выбором отсортированной игры
  - ReplayGameWindow – окно, в котором отображается replay
  - PlayGameWindow – окно с игровым процессом
  - ExtraMenuMindow – дополнительное меню (игра с ботом)
  - MenuMainWindow – главное меню игры

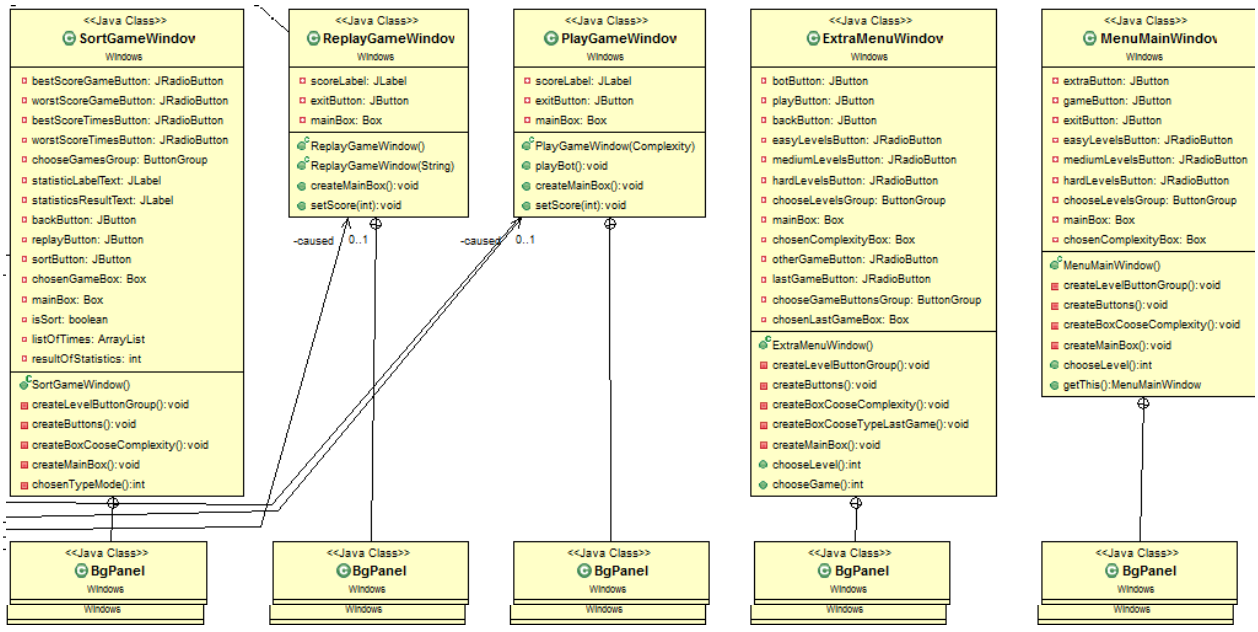
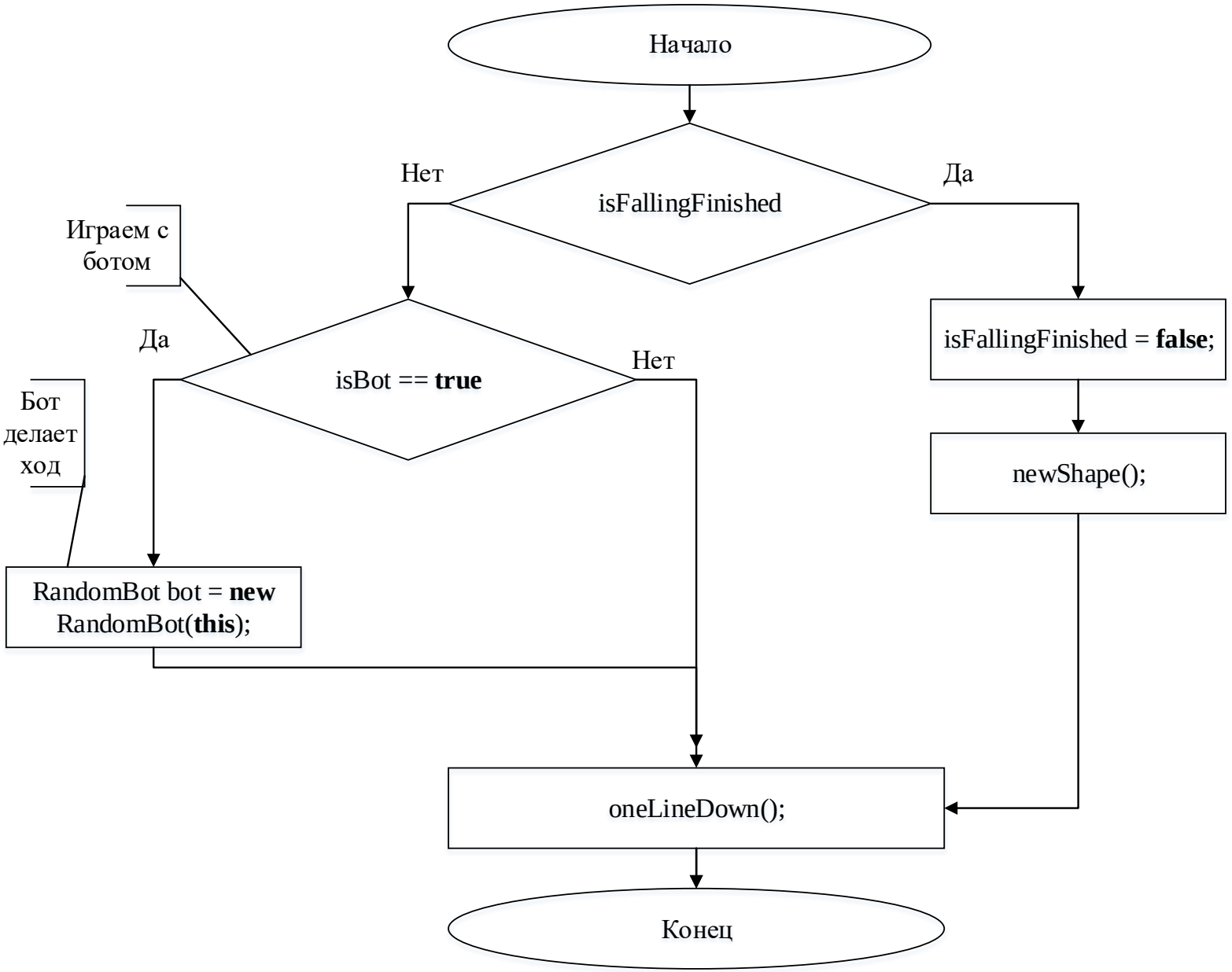


Рис. 8 Пакет Windows

Приложение Б (Блок схема алгоритма функции actionPerformed())



Приложение В (самогенерируема документация)

TetrisLogic

Class Complexity

java.lang.Object
    TetrisLogic.Complexity

```
public class Complexity
extends java.lang.Object
```

Класс, хранящий информацию о уровне сложности

Nested Class Summary

Nested Classes

Modifier and Type	Class and Description
static class	Complexity.typeComplexity Тип выбранной сложности

Constructor Summary

Constructors

Constructor and Description
Complexity(int chosenComplexity) Конструктор, заполняющий поля с информацией об игре, в зависимости от выбранной сложности

Рис. 1

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
int	**getBoardHeight**() Получить высоту игрового поля, определённую по выбранной сложности
int	**getBoardWidth**() Получить ширину игрового поля, определённую по выбранной сложности
int	**getTimeFalling**() Получить время падения фигуры, определённое по выбранной сложности
Complexity.typeComplexity	**getTypeComplexity**() Получить тип выбранной сложностиъ

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Complexity

```
public Complexity(int chosenComplexity)
```

Конструктор, заполняющий поля с информацией об игре, в зависимости от выбранной сложности

Parameters:

chosenComplexity - номер, соответствующий уровню сложности

Рис. 2

Method Detail

getBoardWidth

```
public int getBoardWidth()
```

Получить ширину игрового поля, определённую по выбранной сложности

Returns:

ширина игорового поля

getBoardHeight

```
public int getBoardHeight()
```

Получить высоту игрового поля, определённую по выбранной сложности

Returns:

высота игорового поля

getTimeFalling

```
public int getTimeFalling()
```

Получить время падения фигуры, определённое по выбранной сложности

Returns:

время падения фигуры

getTypeComplexity

Рис. 3

```
public Complexity.typeComplexity getTypeComplexity()
```

Получить тип выбранной сложностиъ

Returns:

choosenTypeComplexity тип выбранной сложностиъ

Рис. 4